Photometrics™

Advanced Camera Operation

ROPER SCIENTIFIC™

BEYOND IMAGING

Roper Scientific, Inc.
3440 East Britannia Drive
Tucson, Arizona 85706
Tel: 520.889.9933
Fax: 520.295.0299

57-039-001 Rev C1

LIMITED WARRANTY — Roper Scientific Analytical Instrumentation

Roper Scientific, Inc. makes the following limited warranties. These limited warranties extend to the original purchaser only and no other purchaser or transferee.

Limited One (1) Year Warranty

Roper Scientific warrants this product against defects in materials or workmanship for a period 1 year after the date of original invoice. During this period, Roper Scientific will repair a defective product or part, without charge to you. You must deliver the entire product to the Roper Scientific factory or, at our option, a factory authorized service center. You are responsible for all transportation and insurance charges to return the product to the service center, and Roper Scientific will be responsible for all transportation charges and insurance to return the product to you. International customers should contact your local manufacturer's representative/distributor for repair information and assistance or visit our technical support page at www.roperscientific.com.

Shutter Warranty

Roper Scientific warrants the standard, factory-installed shutter of all our products that incorporate an integrated shutter for a period of twelve (12) months. This warranty applies to the standard shutter installed in the camera system at the time of manufacture. *Non-standard shutters, SPR (special product request) shutters, and third-party shutter drive equipment carry no warranty expressed or implied.* Roper Scientific will supply, at no cost to the customer, up to one (1) replacement shutter during the warranty period. Roper Scientific will, at Roper Scientific's option, either ship a ready-to-install shutter to the customer site for installation by the customer according to the instructions in the product User Manual or arrange with the customer to return the camera system (or portion of the camera system) to the factory (or factory authorized service center) for shutter replacement by a factory-authorized agent. Responsibility for transportation and insurance charges is described above.

Sealed Chamber Integrity Warranty

Roper Scientific warrants the sealed chamber integrity of all our products for a period of twenty-four (24) months. *Open chamber products carry no warranty to the CCD imaging device expressed or implied.*

Vacuum Integrity Warranty

Roper Scientific warrants the vacuum integrity of all our products for a period of twenty-four (24) months during which we guarantee the detector head will maintain the factory-set operating temperature without the requirement for customer pumping.

Image Intensifier Detector Warranty

All image intensifiers by nature are susceptible to Phosphor and/or Photocathode burn (physical) damage when exposed to high intensity light. Roper Scientific warrants, with the exception of an image intensifier that is found to have a Phosphor and/or Photocathode burn damage (which carries no warranty expressed or implied), all image-intensified products for a period of 1 year after the date of the original invoice. *See the Limited One (1) year warranty terms and conditions above.*

X-Ray Detector Warranty

Roper Scientific warrants, with the exception of the CCD imaging device and fiber optic assembly damaged due to x-ray (which carry no warranty expressed or implied), all x-ray products for a period of 1 year after the date of the original invoice. *See the Limited One (1) year warranty terms and conditions above.*

Software Warranty

Roper Scientific warrants all Roper Scientific manufactured software discs are free from defects in materials and workmanship under normal use for a period of one (1) year from date of original invoice. Roper Scientific does not warrant that the function of the software will meet your requirements or that operation will be uninterrupted or error free. You assume responsibility for selecting the software to achieve your intended results and for the use and results obtained from the software. In addition, during the 12-month limited

warranty the original purchaser is also entitled to receive free version upgrades. Version upgrades supplied free of charge will be in the form of a download from the Internet. Those customers who do not have access to the Internet may obtain the version upgrades on a CD-ROM from our factory for an incidental shipping and handling charge. *See Item 12 in the "Your Responsibility" section of this warranty for more information.*

Owner's Manual and Troubleshooting

You should read the owner's manual thoroughly before operating this product. In the unlikely event that you should encounter operation difficulties, the owner's manual should be consulted before calling the factory for support. If you have consulted the owner's manual and the problem still persists, please contact the appropriate factory for support. *See Item 12 in the "Your Responsibility" section of this warranty for more information.*

Your Responsibility

The above warranties are subject to the following conditions:

1. You must retain your bill of sale (invoice) or provide other proof of purchase.
2. You must notify the factory service center within the first thirty (30) days after you have taken delivery of a defective product or part. With the exception of customers who claim a "technical issue" with the operation of the product or part, all invoices must be paid in accordance with the terms of sale. Failure to pay invoices when due may result in the interruption of your one (1) year limited warranty and/or any other warranty expressed or implied.
3. All warranty service must be made by the factory or, at our option, an authorized service center.
4. Before products or parts can be returned for service the customer must contact the factory and receive a return authorization number (RMA). Products or parts returned for service without a return authorization will be sent back freight collect.
5. These warranties are effective only if purchased from the factory or one of our authorized manufacturer's representatives or distributors.
6. Unless specified as part of the original purchase agreement, Roper Scientific is not responsible for installation, setup, or disassembly at the customer's location.
7. Warranties extend only to defects in materials or workmanship as limited above and do not extend to any product or parts which have been lost or discarded by you; to damage to products or parts caused by misuse in violation of instructions furnished by us; or to units which have had serial numbers removed, altered, defaced, or rendered illegible.
8. At your option after the warranty period has expired, you may contact the factory for repair information and extended warranty plans.
9. Physically damaged units or units that have been modified by a customer are not acceptable for repair in or out of warranty and will be returned as received.
10. All warranties implied by state law or international laws, including the implied warranties of merchantability and fitness for a particular purpose, are expressly limited to the duration of the limited warranties set forth above. With the exception of any warranties implied by state law or international laws, as hereby limited, the forgoing warranty is exclusive and in lieu of all other warranties, guarantees, agreements, and similar obligations of manufacturer or seller with respect to the repair or replacement of any parts. In no event shall Roper Scientific liability exceed the cost of the repair or replacement of the defective product or part.
11. This warranty gives you specific legal rights and you may also have other rights that may vary from state to state and internationally from country to country. Some states and countries do not allow limitations on how long an implied warranty lasts, when an action may be brought, or the exclusion or limitation of incidental or consequential damages, so the above provisions may not apply to you.
12. When contacting us for technical support or service assistance, please refer to the factory of purchase, contact your manufacturer's representative or reseller, or visit our technical support page at www.roperscientific.com.

Table of Contents

Chapter 3. Advanced CCD Theory

Chapter 1. Introduction

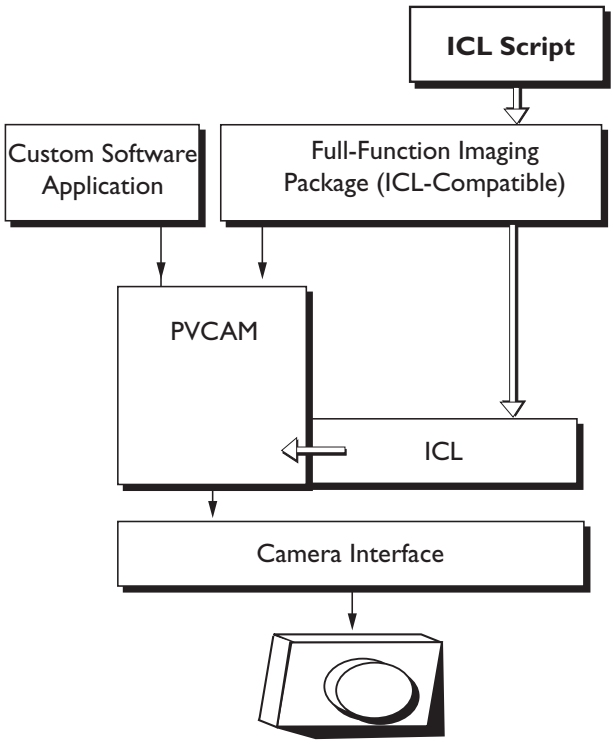
Description

The *Advanced Camera Operation Manual* includes:

- Imager Control Language (ICL)— Rules of syntax, ASCI II command set, and example scripts
- Advanced CCD Theory — Background theory for advanced ICL users

Most Roper Scientific™ cameras use an application programming interface called PVCAM® (Programmable Virtual Camera Access Method). PVCAM is a custom, ANSI C library of camera control and data acquisition functions. Full-function imaging packages access PVCAM. These full-function packages offer the camera control adequate for most users.

ICL is a PVCAM option library that allows users to write low-level, ASCI II command scripts for specialized applications. The scripts, which can be written in any text editor, are then loaded through an ICL-compatible, full–function imaging package, into the ICL option library, then through PVCAM to the camera interface.



Software

To run ICL scripts, you must be running a full-function imaging package that is ICL compatible. You must also have installed the ICL and PVCAM files that are appropriate for your camera and interface.

PVCAM and ICL files are located on the Host Connectivity Kit (HCK) diskette. Installation instructions for these files are covered in your camera system's *Software Guide*.

Roper Scientific Customer Service

If you have any questions regarding your camera system, contact Roper Scientific Customer Service. When you call Roper Scientific, please have your Roper Scientific job number or equipment serial numbers available.

- Phone: 520.889.9933 between 8:00 a.m. and 5 p.m. MST
- Fax: 520.295.0299
- E-mail: cservice@roperscientific.com
- Mail: Roper Scientific
  3440 East Britannia Drive
  Tucson, Arizona 85706

In Europe, you can reach Customer Service at:

BENELUX

- Phone: 31.347.324989
- Fax: 31.347.324979
- E-mail: mailto@roperscientific.com
- Mail: Roper Scientific, BV
  Ir. D.S. Tuijnmanweg 10
  4131 PN VIANEN, Netherlands

FRANCE

- Phone: 33.160.86.03.65
- Fax: 33.160.86.07.09
- E-mail: princeton.instruments@wanadoo.fr
- Mail: Roper Scientific, SARL
  Z.I. Petite Montagne Sud
  4, rue de l'Oisans - C.E. 1702
  91017 Evry Cedex, France

GERMANY

- Phone: 49.89.660.779.3
- Fax: 49.89.660.779.50
- E-mail: mail@roperscientific.de
- Mail: Roper Scientific, GmbH
  Rosenheimer Landstr. 87
  D-85521 Ottobrunn, Germany
-

In Japan, you can reach Customer Service at:

- Phone: 81.43.274.8022
- Fax: 81.43.274.8023
- E-mail: sales@roper.co.jp
- Mail: Nipon Roper, K.K.
  D-10E 1-3 Nakase,
  Mihama-ku, Chiba-shi
  Japan 261-8501

General product information and answers to some customer service questions can be found on our website: http://www.roperscientific.com

Chapter 2. ICL

Introduction

ICL scripts can be written in any text editor. Save the script as a text file, then download through an ICL-compatible imaging package.

This chapter includes:

- Script syntax
- Function descriptions
- Example scripts

Rules of Syntax

The basic rules of syntax are:

- Carriage returns, line feeds, form feeds, tabs, spaces, and comments are treated as generic whitespace used to separate language elements. This convention increases compatibility between operating systems and helps with editors that have different end-of-line conditions and tab expansions.
- There is no main program, subroutines, jumps, calls, conditional statements, or branching.
- Braces are not allowed.
- All numeric values must be typed exactly. Numeric and parenthetical expressions are illegal.

Whitespace

Whitespace includes a single occurrence or any combination of the following: space, carriage return, line feed, form feed, tab, and unnested comment (characters). Whitespace is not required. Once the /* characters are seen, you may insert any desired comments until the closing */ appears. However, you cannot nest comments. The first */ ends the comment. A second /* used before the ending */has no effect, while an additional */ will generate an error.

Parameters / Arguments

Approximately half of the script functions have no parameters. The function must be followed by opening and closing parentheses with no parameters inside the parentheses. Whitespace inside the parentheses is acceptable. Any of the following examples are legal:

```
loop_end( );
loop_end(           );
loop_end(                    );
loop_end( /* comments count as whitespace */ );
loop_end(
);
```

Parameters must be positive integers of normal numeric digits (0...9). None of the following symbols are acceptable:

```
+  -  *  /  .  ,  ^  %  ( )
```

Single Parameter Functions

Some functions require a single parameter. Fractional/decimal and negative values are not allowed. Numeric expressions generate an error. Whitespace can be included anywhere inside the parentheses.

The following parameters are legal:

```
loop_begin(   50   );
loop_begin(     50    /   *exposure count*/     );
loop_begin(   50
);
```

The following parameters are illegal:

```
loop_begin    ( 50 );  whitespace before parentheses
loop_begin(  50,  );   contains a comma
loop_begin(  50  0 );  two numeric entries, only one allowed
loop_begin(  -50  );   minus sign is illegal
loop_begin(  (5*10));  numeric expressions not allowed
```

Multiple Parameter Functions

A few functions require multiple parameters. There are no variable argument lists, so each parameter is always required. The parameters must be separated by commas. Insert whitespace as desired.

The following examples are legal:

```
pixel_readout(0,   100,   1,    50,      2 );
pixel_readout(     0,     100,  1,       50,       2 );
pixel_readout(                    /****MAIN FUNCTION****/
                   0,             /* serial offset of "0" */
                   100,           /* serial size, value:100 */
                   1,              /* serial binning */
                   50,            /* parallel size */
                   2);            /*par bin*/.
```

Verbs

A verb describes which function is performed next. Verb names are a mixture of lowercase text and underscore characters. All verbs are followed by parentheses, even if the verb does not require parameters. There is no whitespace between the verb name and the opening parenthesis. List parameters inside the parentheses and separate the parameters with commas. Whitespace is allowed in the parameter list, but is not required. After the closing parenthesis, add a semicolon. See sample below:

```
verb1();
verb2(parameter, parameter, parameter);
verb3(parameter);
verb4(parameter); (whitespace) verb5();
```

You can use several verbs on a single line, one verb per line, several lines of whitespace, or any combination of the above.

Verbs as Subroutines

You can think of verbs as camera functions or subroutines. A single-verb instruction such as `flash()` or `clear_until_trig()` can be expanded into a sequence of camera-specific instructions. Most verbs are directions for the camera to perform the function immediately using the current settings.

Begin and End

There are two commands required for every script, if either command is missing, the program sets an error code.

```
script_begin();
script_end(contin_clear);
```

`script_begin` specifies the start of the program. Anything appearing before `script_begin` is considered whitespace and is ignored by the script processor. Therefore you can insert comments at the head of a script without using comment delimiters.

`script_end` specifies that the script is finished, transmit to the camera. If `script_end` appears, the script processor will not continue to the null-terminating character at the end of the input string.

Looping Verbs

Use the following verbs for looping:

```
loop_begin(loop_count);
loop_end();
```

All instructions occurring between these two verbs are executed *`loop_count`* times. The mechanism that performs this communication is camera specific. On most systems, built-in commands are used to perform looping. However, on some systems, the instructions inside the loop may be duplicated *`loop_count`* times. You can nest loops, up to 16 deep. For every `loop_begin` function, there must be one `loop_end` instruction. If you create a different number of `loop_begin` and `loop_end`, it generates an error, and the script fails.

Shift Verbs

A shift verb tells the camera to immediately shift one or more lines in the parallel register using the currently selected shifting mode.

A `shift_mode` verb changes the current state of the camera and specifies the clocking method used during exposure (MPP or normal). The new state is implemented the next time that the shift verb is executed.

```
shift_mode_is();
shift_mode_is_alt();
shift_mode_ism();
shift_mode_ism_alt();
shift_mode_s();
shift_mode_s_alt();
shift_mode_sm();
shift_mode_sm_alt();
```

Note that `script_begin` initializes the shift mode to `shift_mode_is`.

Display Verbs

The `pixel_display` verb is not sent to the camera and does not affect data collection. However, once the data has been collected, the application examines the script, the `pixel_display` verbs, and any loop commands. From this information, the application determines how to display the images. (Note that the `pixel_display` function may appear inside loops, outside loops, or both.)

If you use `pixel_readout` anywhere in the script, you must also use `pixel_display`. (If the script does not include `pixel_readout`, `pixel_display` must not appear.)

Syntax Summary

The following is a summary of ICL script syntax:

Script

Every script must start with `script_begin` and end with `script_end`. Match each `loop_begin` with a `loop_end`.

```
Opening comments.
These don't need to be inside comment marks.
script_begin( );
verb( );verb(param,param);
loop_begin(loop_count);
 verb(param);
loop_end( );
pixel_readout(param,param,param,param,param);
pixel_display(param,param);
script_end(param);
```

Whitespace

Whitespace is never required.When whitespace is allowed,the following are allowed:

character:	space	^I	^J	^L	^M	/*…*/(non-nested comments)
ASCII name:	space	HT	LF	FF	CR	
dec. value:	32	9	10	12	13	47,42 … 42,47
C generation:	" "	\t	\n	\f	\r	"/*" … "*/"

Function Syntax

```
verb();
verb_name(param, param, param); whitespace verb(param);
verb(param);
```

- Verb names (functions) are always lowercase. Some functions contain underscores.
- Every verb must be followed by parentheses and terminated by a semicolon.
- The number of parameters is fixed for each verb.
- Parameters must be hard-coded, numeric values, containing only the characters [0…9].
- Parameters must be separated by commas.
- Whitespace may be inserted between parenthesis, commas, parameters, or verbs, but not between verbs and opening parenthesis.

Readout / Display

Neither `pixel_readout` nor `pixel_display` is mandatory, but if one appears, they must both appear. The total number of pixels collected must match the total number of pixels displayed.

Function Definitions

`clear_parallel(clear_count);`	clear the entire parallel register, clear_count times
`clear_serial(clear_count);`	clear the serial register, clear_count times
`clear_until_trig();`	waits for a trigger, clearing meanwhile
`expose(exp_time);`	a timed delay, while light falls on the CCD
`expose_until_trig();`	allow light to fall on the CCD until a trigger pulse
`expose_while_trig(clear_first);`	bulbmode expose while you hold the button
`flash(flash_time);`	activate the flash circuit for flash_time ms
`loop_begin(loop_count);`	loop control, start a loop, do it for loop_count cycles
`loop_end();`	loop control, bottom end of a loop
`pixel_display(x,y);`	instruction to application: display this size image
`pixel_readout(s_offset, s_size, s_bin, p_size, p_bin);`	READ DATA FROM THE CCD this is the only way to output image pixels
`script_begin();`	this must be the first verb in a script
`script_end(contin_clear);`	this must be the last verb in a script if `contin_clear` is non-zero, the CCD is left in continuous clear mode.
`shift(number_of_lines);`	shift the parallel register several lines, using a mode
`shift_image_to_storage( );`	redundant, but useful for frame transfer CCDs
`shift_mode_is( );`	parallel shift mode: image and storage (normal)
`shift_mode_is_alt( );`	parallel shift mode: image and storage (alternate)
`shift_mode_ism( );`	parallel shift mode: image and storage (MPP)
`shift_mode_ism_alt( );`	parallel shift mode: image and storage (MPP alt.)
`shift_mode_s( );`	parallel shift mode: storage array (normal)
`shift_mode_s_alt( );`	parallel shift mode: storage array (alternate)
`shift_mode_sm( );`	parallel shift mode: storage array MPP)
`shift_mode_sm_alt( );`	parallel shift mode: storage array (MPP alternate)
`shutter_close( );`	close the camera shutter
`shutter_open( );`	open the camera shutter

`clear_parallel(clear_count)`
This function clears the parallel register (the entire CCD: the premask area, active area, and postmask area) `clear_count` times where `clear_count` must be greater than zero. The function puts the CCD into an image and storage shifting mode, then shifts the entire parallel register into the serial register, thus clearing the CCD of all charge. This process can also be accomplished by using other functions, such as using a number of shift commands with the proper shift mode, but this function is easier to use. Although the serial register runs continuously during the clearing, there are some circumstances where the serial register may still contain charge. (This condition requires additional clearing with the `clear_serial` command.) Note that this command leaves the parallel shifting mode set to `shift_mode_is`. `clear_count` must be between 1 and 65,535, inclusive.

`clear_serial(clear_count);`
This function clears the serial register (the prescan area, active area, and postscan area) `clear_count` times. The function runs the serial register, dumping any charge into the output node where the charge is transferred into the power supply. `clear_count` must be between 1 and 65,535, inclusive.

clear_until_trig( );
This function causes the CCD to enter clearing mode and continues clearing indefinitely until a trigger arrives. Both the parallel and serial registers are continuously clearing (moving charge toward and into the serial register then out). When the trigger signal arrives, the CCD finishes the current parallel shift (the maximum delay is the time to shift 1 parallel row) and then stops clearing. Execution immediately continues with the next script instruction. For more information concerning the pinouts and electrical specifications of the trigger port, refer to your camera's *User Manual*. (Please note that the current parallel shift is completed before the camera begins integrating.)

expose*(exp_time);*
The CCD exposes for *exp_time* milliseconds. This command usually appears immediately after shutter_open, clear_parallel, or clear_until_trig. Note that exposing is not equivalent to merely waiting for the duration of the exposure time. During an exposure, voltage is applied to the CCD in a gridlike pattern, to produce potential wells (pixels) and to keep the charge from drifting into adjacent cells. On some CCDs, this voltage may be applied in either normal or MPP mode. The mode actually used for the exposure reflects the most recently set parallel shifting mode (whether that mode was set to normal or MPP). *exp_time* may be set to zero, although this instruction becomes a "no operation" under those conditions. The *exp_time* variable maintains full precision up to a value of 65,535 milliseconds. After that point, the program creates longer times through an internal loop counter that provides more precise timing than the scripting loop_begin command. Due to this mechanism, some values may be rounded off to a nearby value (for example, prime numbers greater than 65,535 won't be exactly represented). The maximum exposure length is 2^32 milliseconds (about 49 days). For more information on using exposure in conjunction with frame transfer, consult your camera's *User Manual*.

expose_until_trig( );
This function begins exposing immediately and continues exposing until a trigger signal arrives. The script then continues with the next instruction. During the exposure, this applies voltage patterns to the CCD in either normal or MPP modes (see expose). As with the regular expose function, this neither opens or closes the shutter. For more information concerning the pinouts and electrical specifications of the trigger port, refer to your camera's *User Manual*.

expose_while_trig*(clear_first);*
This function initiates a scripted version of the high-level bulb-mode exposure. If *clear_first* is 1, this initially enters a continuous clearing mode, exactly like clear_until_trig. If *clear_first* is 0, the CCD exposes while it waits for the trigger. In either case, once the trigger arrives, the CCD switches into an exposing mode (either normal or MPP, see expose for more information), and continues exposing while the trigger signal is present. As soon as the trigger ends, the exposure stops, and the script proceeds with the next instruction. Note that this instruction is not redundant. You can't use expose_until_trig in conjunction with other instructions (such as clear_until_trig) to duplicate the function of this trigger, because a trigger signal stops the exposure with expose_until_trig, while that same trigger signal starts (and maintains) expose_while_trig. For more information concerning the pinouts and electrical specifications of the trigger port, refer to your camera's *User Manual*. clear_first must be either 0 or 1.

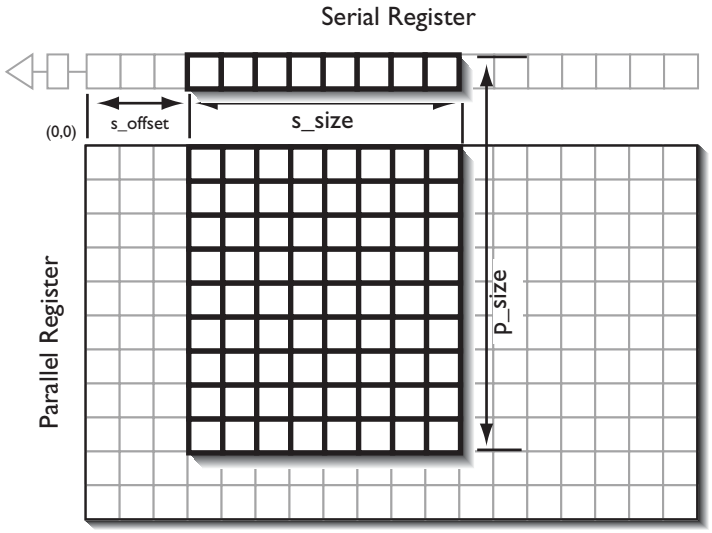
flash*(flash_time);*
This function activates the flash circuit (a set of pins on the trigger port of many PVCAM cameras) and continues applying power for *flash_time* milliseconds. In some cases, this may be used to activate devices connected to the camera, such as an illuminator or filter wheel (although the flash signal differs from camera to camera and is often not a TTL-compatible signal). For more information concerning the pinouts and electrical specifications of the trigger port, refer to your camera's *User Manual*. *flash_time* must be between 1 and 65,535, inclusive.

loop_begin*(loop_count);*
This function allows looping within a script. A *loop_count* specifies the number of times to perform the loop. All instructions between the loop_begin and matching loop_end commands are executed exactly *loop_count* times. Loops may be nested up to 16 deep. Note that indentation of a script's source code may improve readability, but it does not alter the loop nesting in any way. *loop_count* must be between 1 and 65,535, inclusive.

loop_end( );
This function defines the end-of-loop, and allows looping within a script. This command must be matched with a loop_begin command (the loop_begin command must appear first). Loops can be nested.

pixel_display*(x,y);*
This function indicates that camera output is decoded for display on a monitor. The application software takes the next (*x*y*) pixels from the output data stream and displays them as a single rectangular image, x pixels wide by y pixels tall. Cross-checking done during script setup ensures that the total number of pixels displayed matches the total number of pixels collected (assuming that data collection is completed without errors). In other words, the total number of pixels read from the camera (using pixel_readout) equals the total number of pixels displayed (using pixel_display). Depending on the experiment design and the exact script used, the individual pixel_readout and pixel_display instructions may or may not be closely matched. However, if the script contains any pixel_readout instructions (i.e., if one or more pixels are readout from the camera), it must also contain at least one pixel_display instruction. Both x and y must be between 1 and 65,535, inclusive.

pixel_readout(s_offset, s_size, s_bin, p_size, p_bin);
This function causes a block of pixels (region) to be first read out into the serial register, then transferred into the output converter and digitizer. The region must be immediately adjacent to the serial register when this instruction is given (the parallel offset must be zero, so you have to use the shift command to move the desired region to the edge of the parallel register).



For each row of the block, the first *s_offset* pixels are skipped. The next *s_size* pixels (after the skipped pixels) are digitized using a binning of *s_bin*. Each subsequent row is then digitized in the same fashion for a total of *p_size* rows. Finally, if *p_bin* is greater than 1, parallel binning is also performed.

All parallel shifting is performed using the current parallel shifting mode. In some cases, the resulting readout makes no sense (for example, if a custom backward shift is used).

If any of the sizes are an uneven multiple of binning, a smaller size that exactly fits the binning is used. If the size is smaller than the binning (size 4, binning 5) a fatal error is produced.

You can use this function to stack several regions, one after the other, in the parallel direction. However, you cannot stack more than one region at a time in the serial direction. *s_offset* may be zero. All other parameters must be between 1 and 65,535, inclusive. *s_size* and *p_size* must be at least as large as their corresponding *s_bin* and *p_bin*. Finally, *s_offset* and *s_size* must be no larger than the CCD serial size.

script_begin( );
This must be the first instruction in the script. It signals that the script is starting now. Any text that occurs before this instruction is ignored. This allows you to put in an initial comment block that can be used to explain the purpose and operation of script programs. This instruction automatically puts the CCD into shift_mode_is.

script_end(*contin_clear*);
This must be the last instruction in a script. It signals to the compiler that the script program is now finished. Any text that occurs after this instruction is ignored. If the parameter *contin_clear* is 1, the camera remains in continuous clear mode. This indefinitely cycles the CCD in a shift-and-eliminate-charge loop, similar to the clear_until_trig instruction. Since this actively cycles power through the CCD, it also generates heat within the CCD. This may be a problem in some cases, particularly if the camera is run near the low-temperature limit. Continuous clearing occurs until a new script or exposure is started, an abort is sent, or until the camera hardware is reset or turned off. Other commands to the camera (such as altering the speed setting) also cancel continuous clearing. *contin_clear* must be either 0 or 1.

shift(*number_of_lines*);
This function shifts the *number_of_lines* rows of data, in the parallel direction, using the current shift mode. Depending on the shift mode in use, this may or may not shift the entire parallel register (it may shift only the storage array), shift the rows either forward or backward (depending on the setting of the ALT shift modes), or use MPP mode for the clocking. The serial register is cleared during the shift operation, so any charge dumped into the serial register is eliminated. The two most common cases are described below:

> **shift_mode_is** or **shift_mode_ism**: In these two modes, the entire parallel register is being moved. Issuing the instructions shift(3); moves the entire parallel register 3 rows closer to the serial register. The far end of the parallel register is filled with zeros (no charge). The three rows closest to the serial register are dumped in to the serial register and cleared.
>
> **shift_mode_s** or **shift_mode_sm**: These two modes can only be performed on frame-transfer devices. Issuing the instruction shift(3); moves the storage array three rows closer to the serial register. The far end of the storage array is filled with zero (no charge). The image array is completely unaffected (it is often left exposing). The three rows closest to the serial register are dumped into the serial register and cleared.

Please note that the *alternate* shift modes are usually loaded at the factory with settings that are identical to the normal modes. You can request custom settings that allow backward shifting, shift image only, etc.

Also note that shifting is not useful for outputting pixels. It is useful only for moving a region into position for readout. Readout must be done through the pixel_readout command. *number_of_lines* must be between 1 and 65,535, inclusive.

shift_image_to_storage( );
This function can only be used on frame-transfer devices. It shifts the CCD's image array into the storage array, and any data currently in the storage area is lost. (It is shifted into the serial register, and the serial register is then cleared.) Although this operation could also be accomplished using appropriate combinations of the shift instruction and various shifting modes, this function does the operation more efficiently with a single instruction. Please note that using this command leaves the parallel shifting mode set to shift_mode_s.

shift_mode_is( );
Shift Image and Storage. This is a parallel shifting mode that shifts the entire parallel register. This mode can be used on all CCDs and uses the normal clocking method that is equivalent to MPP off. The exception is for CCDs that require MPP clocking, and, in this case, MPP on is used.

shift_mode_is_alt( );
Shift Image and Storage, Alternate. This is a parallel shifting mode that shifts the entire parallel register. This mode can be used on all CCDs and uses the normal clocking method that is equivalent to MPP off. The exception is for CCDs that require MPP clocking, in this case, MPP on is used. In addition to the MPP style of shifting, this function uses an alternate shifting mode (see `shift`). Unless an alternate shifting was loaded at the factory, this mode is loaded with a default value identical to `shift_mode_is`.

shift_mode_ism( );
Shift Image and Storage, MPP. This is a parallel shifting mode that shifts the entire parallel register. This mode can only be used on CCDs that support MPP clocking. If this mode is used on a non-MPP CCD, the script compiler generates an error and fails.

shift_mode_ism_alt( );
Shift Image and Storage, MPP, Alternate. This is a parallel shifting mode that shifts the entire parallel register. This mode can only be used on CCDs that support MPP clocking. If this mode is used on a non-MPP CCD, the script compiler generates an error and fails. In addition to using MPP clocking, this function uses an alternate shifting mode (see `shift`). Unless an alternate shifting was loaded at the factory, this mode is loaded with a default value identical to `shift_mode_ism`.

shift_mode_s( );
Shift Storage. This parallel shifting mode shifts only the storage array on a frame-transfer CCD. This mode can only be used on frame-transfer CCDs. If this mode is used on a non-frame-transfer CCD, the script compiler generates an error and fails. This function uses the normal clocking method that is usually equivalent to MPP off. For CCDs that require MPP clocking, the clocking method is MPP on.

shift_mode_s_alt( );
Shift Storage Alternate. This parallel shifting mode shifts only the storage array on a frame-transfer CCD. This mode can only be used on frame-transfer CCDs. If this mode is used on a non-frame-transfer CCD, the script compiler generates an error and fails. This clocking is equivalent to MPP off except for CCDs that require MPP clocking. In this case, the clocking would be equivalent to MPP on. In addition to the MPP style of shifting, this function uses an alternate shifting mode (see `shift`). Unless an alternate shifting was loaded at the factory, this mode is loaded with a default value identical to `shift_mode_s`.

shift_mode_sm( );
Shift Storage, MPP. This parallel shifting mode shifts only the storage array on a frame-transfer CCD. This mode can be used on CCDs that support both MPP clocking and frame transfer. If this mode is used on a non-MPP or non-frame-transfer CCD, the script compiler generates an error and fails.

shift_mode_sm_alt( );
Shift Storage, MPP, Alternate. This parallel shifting mode shifts only the storage array on a frame-transfer CCD. This mode can be used only on CCDs that support both MPP clocking and frame transfer. If this mode is used on a non-MPP or non-frame-transfer CCD, the script compiler generates an error and fails. In addition to the MPP style of shifting, this function uses an alternate shifting mode (see `shift`). Unless an alternate shifting was loaded at the factory, this mode is loaded with a default value identical to `shift_mode_sm`.

shutter_close( );
This function closes the shutter. Once the *`shutter_close_delay`* time has passed, this function continues on to the next script instruction. Since there is no default shutter position, and if the shutter position is important to the current script, set the shutter open or closed at the start of the script program. Note that none of the other script instructions (except for `shutter_open`, of course) alters the shutter state. Specifically, none of the `expose`

commands affect the shutter. For more information concerning the pinouts and electrical specifications of the trigger port, and how they relate to the shutter commands, refer to your camera's *User Manual*.

`shutter_open( );`
This function opens the camera shutter. Once the *`shutter_open_delay`* time has passed, this function continues on to the next script instruction. See `shutter_close` for more information. For more information concerning the pinouts and electrical specifications of the trigger port, and how they relate to the shutter commands, refer to your camera's *User Manual*.

Example Scripts

The eight example scripts in this section illustrate the rules and principles of ICL. These scripts use every ICL function with the exception of shift_mode_sm() and the _alt modes. Electronic copies of the example scripts are in the ICL Example file on your HCK diskette.

- Open the Shutter
- Single Image
- Time Delayed Integration Panorama
- Ratio Imaging: 2-Frame Ratio
- Ratio Imaging: Multi-Frame Ratio
- 3-Color Sequence
- Intermittent Exposure
- High-Speed Spectroscopy

Open the Shutter
This script opens the camera shutter and leaves it open.

```
script_begin();
shutter_open();
script_end(0);
```

Single Image
This script is intended for use on a Kodak 1400 CCD (serial size 1317, parallel size 1035). The script takes one full-frame image and exits.

```
script_begin();
shutter_close();                  /* good precaution—who knows what state it's in    */
clear_parallel(2);                /* clear out any residual charge                   */
clear_serial(2);                  /* might as well clear this out, too               */
shutter_open();                   /* START TAKING THE IMAGE...                       */
 expose(200);                     /* expose for 200 milliseconds                     */
shutter_close();                  /*...DONE TAKING THE IMAGE                         */
pixel_readout(0,1317,1,1035,1);   /* readout the entire CCD                          */
pixel_display(1317,1035);         /* display the entire CCD                          */
script_end(1);                    /* leave the CCD in continuous clear mode          */
```

TDI (Time Delay Integration) Panorama

This script is written for use with a Kodak 1400 CCD (serial size 1317,parallel size 1035). The camera is attached to a telescope pointing straight up. The camera's external trigger port is connected to a synchronized time source that provides a pulse every few seconds. The pulse timing is closely coordinated with the image size, so the rotation of the earth moves the image on the CCD by exactly one parallel row width (6.8 microns) for each pulse. Therefore, for every pulse, one row is readout, and the other rows are shifted over, following the moving image. This script uses the setup to collect a single panoramic image 10,000 rows wide (reading 1 row at a time, 8965 times, and finishing with a final clean up readout of 1035 rows).

The CCD reaches a steady state exposure duration during the middle of the panorama, but the beginning and ending few lines are exposed for less time. (Exposure time is progressively less for the first and last 1035 rows of the panorama.) The result is that the front and back ends of the panorama gradually ramp up and down in brightness.

```
script_begin();
shift_mode_is();                /* redundant, since script_begin sets this anyway*/
shutter_open();                 /* light is now falling on the CCD               */
clear_parallel(5);              /* make sure this CCD is cleared out             */
clear_until_trig();             /* clear until acquire regular pulses            */
loop_begin(8965);               /* 10,000 (full panorama) - 1035 (clean-up size) */
  expose_until_trig();          /* readout each time trigger pulse received      */
  pixel_readout(0,1317,1,1,1);  /* read exactly 1 row, shift the rest over       */
loop_end();
pixel_readout(0,1317,1,1035,1);/*clean up: read the entire CCD                   */
pixel_display(1317,10000);      /* display this as one huge panoramic image      */
shutter_close();
script_end(0);
```

Ratio Imaging: 2-Frame Ratio

This script collects two frames in quick succession, for use in a ratio imaging experiment. The camera is a PXL®/37 (a Marconi CCD37-10 CCD, with a 512x512 image area, and a masked 512x544 storage area). Since the exposures must be tightly coordinated with the experimental setup, the camera and experiment communicate with each other as follows:

1. The script is set up, and the camera is started (note, the camera shutter opens, but it continuously clears until step 3).
2. The final experimental setup is performed, including turning on the primary lighting.
3. The experimental hardware provides a rising-edge TTL signal when it is ready for the first image to be taken.
4. Upon receipt of the TTL signal, the camera exposes the first image, and then stores it.
5. The camera begins the second exposure. Simultaneously, it provides a flash signal (see the *PXL User Manual*, this closes the ground side of a 10mA +15VDC current source).
6. Upon detection of the flash signal, the experimental hardware switches off the primary lighting and turns on the secondary lighting.
7. The camera finishes the exposure, and turns off the flash signal. It then begins to close the shutter.
8. When the flash signal stops, all lighting turns off, too (this compensates for the relatively slow mechanical shutter).
9. The camera outputs data for both exposures.

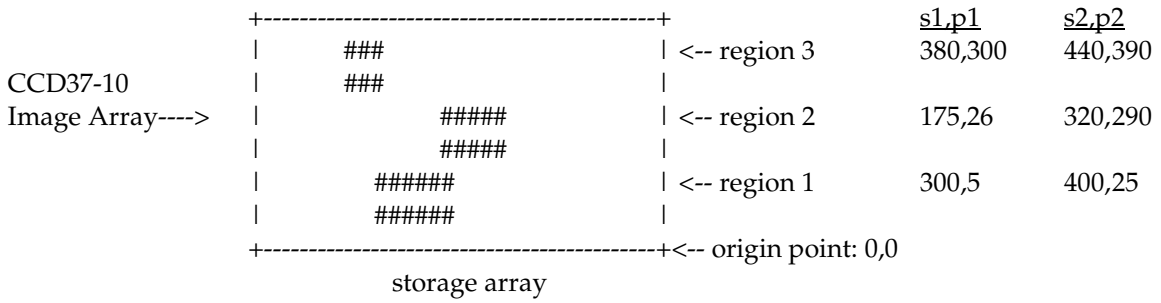
This script takes two very fast exposures (10 milliseconds each) with minimal delay between each exposure (on the CCD37-10, shifting an image into storage takes less than 1.3 milliseconds). The shutter is slower than the exposures. (A typical mechanical shutter requires 15 milliseconds to close.) To remove this effect, lighting must be shut off as soon as the second exposure is finished. It is assumed that the lights instantly drop to zero intensity.

```
script_begin();
shutter_open();                 /* STEP 1:   light is falling on the CCD           */
clear_parallel(2);              /*           make sure the CCD is cleared          */
clear_until_trig();             /*           clear until the experiment is ready   */
expose(10);                     /* STEP 4:   take a 10 millisecond exposure        */
shift_mode_is();                /*           prepare to shift image into storage   */
shift(512);                     /*           shift image into top of storage       */
flash(10);                      /* STEP 5:   a 10 ms exposure, send flash signal   */
shutter_close();                /* STEP 7:   begin to close the shutter            */
shift(32);                      /* STEP 9:   discard useless front end of storage */
pixel_readout(0,512,1,          /*           output both exposures (in one command)*/
 1024 ,1 );                     /*           (two 512 exposures = 1024 rows)      */
pixel_display(512,512);         /* Display first image                             */
pixel_display(512,512);         /* Display second image                            */
script_end(0);
```

IMPORTANT NOTE: The command `shift_image_to_storage` leaves the CCD set to `shift_mode_s`. If a readout was done without setting the proper shift mode, the first exposure would read properly, then read out 512 blank, cleared lines, leaving the image array untouched.

Ratio Imaging: Multi-Frame Ratio

This script performs ratio imaging with the CCD37-10 CCD. Only three small subregions are needed, but to get a good statistical base for the images, you must take a sequence of 100 identical exposures. The subimage setup is:



These coordinates are INCLUSIVE. 300,5 to 400,25 is a region of 101 serial columns and 21 parallel rows. This experiment swings in a filter (relatively slowly), sends a trigger pulse, exposes for 100 milliseconds, reads out, then repeats.

H/W A:	experimental hardware goes to first position, sends trigger pulse,
H/W B:	the first position runs for 100 milliseconds,
H/W C:	change filters (about 40 milliseconds of delay)
H/W D:	the second position runs for 100 milliseconds
H/W E:	change filters (about 40 milliseconds of delay), loop back to A
CCD A:	prepare for exposure, clear CCD, wait for trigger...
CCD B:	expose for 100 milliseconds
CCD C:	`shift_image_to_storage`, readout, clear CCD, wait for trigger...
CCD D:	expose for 100 milliseconds
CCD E:	`shift_image_to_storage`, readout, loop back to A

The only problem is that you cannot use a signal to differentiate between CCD B and CCD D. In the future, camera hardware might be enhanced so that it produces a flash signal in CCD E. Then you would use the enhancement to trigger step H/W A.

```
script_begin();
shutter_open();
loop_begin(200);                    /* SEQUENCE OF 100 IMAGE PAIRS(200 total loops)  */
  clear_parallel(4);                /* CCD A&C: make sure CCD is cleared              */
  clear_until_trig();               /*          clear until the experiment is ready  */
  expose(100);                      /* CCD B&D:  expose with this filter              */
shift_image_to_storage();           /* takes 1.23 msec, shift image under mask       */
shift_mode_s();                     /* CCD C&E: READOUT...                           */
 shift(37);                         /*          move REGION 1 up to serial register  */
 pixel_readout(300,101,1, 21,1);    /*          s1:s2 of 300:400 is 101 total pixels */
 pixel_display( 101, 21 );          /*          actual size of the region            */
 pixel_readout(175,146,1,265,1);    /*          REGION 2 is at the serial reg        */
 pixel_display( 146, 265 );
 shift(9);                          /*          move REGION 3 up to the serial regist*/
 pixel_readout(380, 61,1,265,1);
 pixel_display( 61, 265 );
loop_end();
shutter_close();
script_end(1);                      /* leave CCD in continuous clear mode            */
```

NOTE: The CCD37-10 has a 512x512 image array, but a 512x544 storage array. Discard (5+32) rows to move region 1 to the serial register.

3-Color Sequence
In this setup the target is illuminated by three filtered lights. Each light can be switched on and off almost instantaneously (in less than a millisecond). There is a small amount of camera-triggered hardware logic that coordinates the hardware with the exposure. All lights are turned off when the Shutter Closed Output pin is on. The experiment resets and starts again when the Shutter Open Output pin goes high (and the first light turns on). Then when the camera's Trigger Waiting Output pin goes high, the experiment switches to the next light (cycling: red-green-blue).

The trigger port is being jumpered into itself. In other words, the Trigger Waiting Output is jumpered back into the Trigger Level Input. (The Trigger Invert Level Input is jumpered to Ground.) The final result is that (in the script below) the camera begins exposing while it waits for a trigger, then, it triggers itself. The Trigger Waiting Output pin has a high signal on it for about 7 microseconds. When you see the high signal, switch to the next light in the sequence (red-green-blue-red-green-blue-etc.).

All of the pins are standard TTL voltage levels, so the associated circuitry is fairly straightforward.

It is important to take this sequence as FAST as possible, so, use the CCD37-10 frame-transfer CCD (512x512 image size, 512x544 storage size) and readout one exposure, while exposing the next. Although the script doesn't appear to expose at all (aside from the first image), the exposure time is equivalent to (readout time + shift time), or about (140 + 1.2) 141 milliseconds.

```
script_begin();
shutter_open();                 /* SIGNAL TO BEGIN THE EXPERIMENT                  */
clear_parallel(6);              /* clear CCD of charge: 2.54 ms / clear            */
expose(141);                    /* give proper exposure time for 1st image         */
loop_begin(50);                 /* SEQUENCE OF 50 IMAGE TRIOs                      */
    loop_begin(3);              /* one trio                                        */
      shift_image_to_storage(); /* 1.23 millisec, leaves mode: shift_mode_s        */
      expose_until_trig();      /* CHANGE TO NEXT LIGHT (7 micro-sec pulse)        */
      shift(32);                /* discard excess in CCD37-10 storage array        */
      pixel_readout(0,512,1,512,1);/* about 140 milli-sec at a speed of 2 MHz      */
      pixel_display( 512, 512 );/* this has no effect on the expose timing         */
      loop_end();
    loop_end();
shutter_close();                /* SIGNAL EXPERIMENT END & RESET                   */
script_end(1);                  /* leave the CCD in continuous clear mode          */
```

Intermittent Exposure
This script takes a stellar image of a faint galaxy. The observatory is located under the flight path of a local airport and incoming planes occasionally pass through the field of view. The observatory has at least 30 seconds warning before a plane passes over, so the camera needs a signal to close the shutter, stop exposing (temporarily), then begin exposing again. The camera supports MPP mode, so the dark current can be drastically reduced during the several-hour exposure.

A switch, connected to the camera's trigger input pin, controls when the shutter opens and closes. (The switch is simply an open/close switch inserted between the ground and the trigger pin on the camera's trigger port.) When the switch is turned on, the camera opens the shutter and exposes. When the switch is turned off, the shutter closes and the exposure waits. (The camera is still exposing, but it has a closed shutter and minimal dark current.)

```
script_begin();
shutter_close();                  /* good precaution—who knows what state it's in     */
clear_parallel(5);                /* clear out any residual charge                    */
clear_serial(2);                  /* might as well clear this out too                 */
shift_mode_ism();                 /* force the CCD to shift and EXPOSE in MPP mode    */
clear_until_trig();               /* wait for the trigger to be turned on...          */
loop_begin(100);                  /* this allows exactly 99 interruptions             */
   expose_until_trig();           /* shutter is closed, waiting for a trigger         */
   shutter_open();                /* open the shutter and begin exposing              */
   expose_while_trig(0);          /* GOOD EXPOSURE TIME                               */
   shutter_close();               /* interruption...                                  */
loop_end();
clear_serial(100);                /* an evening’s worth of charge has accumulated     */
pixel_readout(0,1024,1,1024,1);   /* readout the entire CCD (1024x1024)               */
pixel_display(1024,1024);         /* display the entire CCD                           */
script_end(0);                    /* DO NOT CLEAR the CCD                             */
```

There is one problem. If there have only been a half-dozen interruptions, how do you finish the readout? The best way is to turn the switch off (which closes the shutter), put on a lens cap, and toggle the switch about a hundred more times. A somewhat riskier method would be to:

- Turn the switch off, thus closing the shutter
- Use the application to ABORT this script (which shouldn't erase the image)
- Run the next script...

```
script_begin();                   /* FINISH READOUT OF A 1024x1024 CCD                */
shutter_close();                  /* just to make sure                                */
clear_serial(100);                /* DO NOT clear the parallel register               */
shift_mode_ism();                 /* do all shifting in MPP mode                      */
pixel_readout(0,1024,1,1024,1);   /* readout the entire CCD (1024x1024)               */
pixel_display(1024,1024);         /* display the entire CCD                           */
script_end(0);
```

This last script needs to be typed into its own file, since everything after the script_end instruction (including that second script) is treated as a comment.

High-Speed Spectroscopy

This script uses a frame-transfer device (the CCD37-10 CCD) to perform high-speed spectroscopy. It collects a spectral image in the image array, and bins that image down to a single line, as it shifts it into the storage area. THIS REQUIRES A FACTORY MODIFICATION TO THE CAMERA (SEE BELOW). The camera immediately performs another exposure, bins it, etc.

A total of 544 binned spectra are accumulated in the storage array, then read out. (Although the CCD37-10 image array is 512x512, the storage array is 512x544.) Each spectral line has a total exposure time of about 1.23 milliseconds, so this gives 2/3 of a second of continuous, high-speed spectral monitoring, with nearly a millisecond time resolution.

```
script_begin();
shutter_open();
clear_parallel(2);              /* this clears both image and storage              */
expose(1);                      /* compensate for the first exposure time          */
loop_begin(544);                /* LOOP FOR EACH SPECTRUM / EACH ROW IN STORAGE    */
   expose(0);                   /* total exposure time: (0 + 1.23) milliseconds    */
   shift_mode_s();              /* shift the storage array only                    */
   shift(1);                    /* move storage down 1 line, to make way for...    */
   shift_mode_s_alt();          /* custom setting: shift IMAGE ARRAY ONLY          */
   shift(512);                  /* bin entire image array into top of storage      */
                                /* NOTE: this shift takes 1.23 milliseconds        */
loop_end();
shutter_close();                /* DONE EXPOSING, NOW READOUT                      */
pixel_readout(0,512,1,544,1);   /* read out the entire storage array               */
loop_begin(544);
   pixel_display(512,1);        /* display data as 512 separate lines              */
loop_end();
script_end(0);
```

FACTORY MODIFICATION: For high-speed spectroscopy, the CCD must be able to shift ONLY the image array. (This accomplishes binning where charge accumulates on the top row of the storage array.) In this case, the new shift table is loaded into one of the alternate shifting modes. Specifically, `shift_mode_s_alt()` is reloaded, so it performs: A parallel shifting mode, this shifts the IMAGE array only on a frame-transfer CCD. It should be called shift_mode_I_alt(), but that choice is not available.

Error Codes

This section lists the error codes that may appear when using ICL.

10100	C101_ICL_UNKNOWN_ERROR	ICL OPTION LIBRARY: unknown error
10101	C101_ICL_LIB_NOT_INIT	the script library not initialized
10102	C101_ICL_LIB_INITED	the script library initialized
10103	C101_ICL_NO_BEGIN	script_begin command was not seen
10104	C101_ICL_END_TOO_SOON	text ended before script_end instruction
10105	C101_ICL_INST_INVAL	script instruction not correctly read
10106	C101_ICL_OPEN_PAREN	opening parenthesis not present
10107	C101_ICL_ILLEGAL_CHAR	illegal character or symbol
10108	C101_ICL_BAD_COMMA	unexpected comma
10109	C101_ICL_BAD_NUMBER	unexpected numeric parameter; comma needed?
10110	C101_ICL_BAD_CL_PAREN	unexpected closing parenthesis; extra comma?
10111	C101_ICL_NO_SEMICOLON	semicolon missing from this instruction
10112	C101_ICL_TOO_MANY_ARG	instruction has too many parameters
10113	C101_ICL_TOO_FEW_ARG	instruction doesn't have enough parameters
10114	C101_ICL_ARG_IS_ZERO	argument must be greater than zero
10115	C101_ICL_ARG_OVER_65K	argument must be 65,535 or less
10116	C101_ICL_ARG_INVALID	argument is invalid or illegal
10117	C101_ICL_OVER_LOOP	loops are nested too deeply
10118	C101_ICL_UNDER_LOOP	too many loop_end instructions
10119	C101_ICL_UNEVEN_LOOP	loop_begin commands don't match loop_end
10120	C101_ICL_BIN_TOO_LARG	readout's binning exceeds its size
10121	C101_ICL_RGN_TOO_LARG	readout region does not fit on the CCD
10122	C101_ICL_DISPLAY_SMAL	displayed data is less than the collected data
10123	C101_ICL_DISPLAY_LARG	displayed data is more than the collected data
10124	C101_ICL_NO_FRAME_XFR	camera doesn't have a separate storage array
10125	C101_ICL_NO_MPP	camera does not allow MPP mode
10126	C101_ICL_TOO_COMPLEX	script exceeds available program memory

Man Pages

This section contains the Man Page descriptions for the six ICL scripting functions included in the PVCAM library. These functions are intended for use by application programmers only.

PVCAM **Class 101: ICL** **pl_exp_display_script(101)**

NAME

`pl_exp_display_script` – lists the display rectangles.

SYNOPSIS

```
boolean
   pl_exp_display_script(int16 hcam, icl_disp_ptr
                user_disp_array, void_ptr pixel_stream)
```

DESCRIPTION

This function can only be called after `pl_exp_setup_script`. It further processes the script that was loaded during `pl_exp_setup_script`. Users must pass in a structure that has at least *num_rects* elements *(num_rects* is passed back from `pl_exp_setup_script`). This function then fills that structure with the x and y sizes for every "display" rectangle, as well as that rectangle's offset into the pixel stream (which is why the allocated data collection pointer must be passed in).

RETURN VALUE

TRUE for success, FALSE for a failure. Failure sets pl_error_code.

SEE ALSO

`pl_exp_setup_script(101)`

NOTES

The `script_disp_ptr` is defined in `pv_icl.h`:

```
typedef struct { /* ONE IMAGE "DISPLAY" FOR SCRIPTING */
   uns16 x; /* image width to display, in pixels */
   uns16 y; /* image height to display, in pixels */
   void_ptr disp_addr; /* starting address for this image */
} icl_disp_type,
PV_PTR_DECL icl_disp_ptr;
```

The list of rectangles is unrolled from inside the looping constructs, so users are presented with a simple linear list. `disp_addr` is the starting address for the data for this rectangle (based on the fact that all the data is put into the address *`pixel_stream`*. In other words: the address of the first rectangle is exactly *`pixel_stream`*. If that rectangle was a 10x10 display, the (starting) address of the second rectangle would be *`pixel_stream`* + 200 (bytes).

Separate rectangle lists are kept for each *`hcam`*, so multiple cameras (and multiple users) do not collide. This reports on the state following the most recent call to `pl_exp_setup_script( )` using this value of *`hcam`*. A new call resets the list. This function can be called after data collection has finished, provided that a new call to `pl_exp_setup_script( )` is not made in the mean time.

This function does not actually display data. It only provides display parameters.

NAME

`pl_exp_init_script` – initialize the scripting library.

SYNOPSIS

```
boolean
   pl_exp_init_script(void)
```

DESCRIPTION

This function prepares and initializes the scripting option library. It must be called once, before any other scripting functions are called.

RETURN VALUE

TRUE for success, FALSE for a failure. Failure sets pl_error_code.

SEE ALSO

`pl_pvcam_init(2), pl_pvcam_uninit(2), pl_exp_uninit_script(101)`

NOTES

This function must be called explicitly after calling `pl_pvcam_in`it and before calling any other `pl_exp_ ... _script` function.

Scripting requires some of the functions in the Class 3 PVCAM library. When scripting is initialized, the Class 3 library must also be initialized (`pl_exp_init_seq( )`). Order is not important, so either the scripting or initialization function can be initialized first.

NAME

`pl_exp_listerr_script` – if an error occurred during script processing, list the exact position of that error.

SYNOPSIS

```
boolean
  pl_exp_listerr_script( int16 hcam, char_ptr err_char,
                uns32_ptr err_char_num, uns32_ptr err_line,
                uns32_ptr err_ch_in_line )
```

DESCRIPTION

By default, this function returns zeros for all values, indicating that no logical or syntactical error was spotted in the script. This function resets each time `pl_exp_setup_script` is called. If the last call to `pl_exp_setup_script` generates an error, this function returns the location (where processing stopped) of that error in the script string. In some cases, the location reported may just be past the location of the actual error.

The character that generated the error is returned in *`err_char`*. This will be character number *`err_char_num`* in the input string (0-indexed, as with all C strings). The script is also examined for line-feed, newlines, etc. in an attempt to divide the script into the separate lines used by common word processors. The line number and character in the line are returned in *`err_line`* and *`err_ch_in_line`*. Both of those values are 1-indexed, as is typical with text editors and word processors, so that the first character in the script is character 1 of line 1.

RETURN VALUE

TRUE for success, FALSE for a failure. Failure sets pl_error_code.

SEE ALSO

`pl_exp_setup_script(101)`

NOTES

Separate error lists are kept for each *`hcam`*, so multiple cameras (and multiple users) will not collide. This reports on the error state following the most recent call to `pl_exp_setup_script( )` using this value of *`hcam`*. A new call to `pl_exp_setup_script( )` resets the error list.

Some errors indicate a problem with the entire program rather than a single line (for example, errors 10122 or 10123). In such a case, the error will be listed as occurring at “character 0, line 0.” This indicates that while no single line is at fault, there was an inconsistency in the program at large.

NAME

`pl_exp_setup_script` – process a script; download instructions to the camera.

SYNOPSIS

```
boolean
   pl_exp_setup_script (int16 hcam, char_const_ptr script,
                        uns32_ptr stream_size, uns32_ptr num_rects)
```

DESCRIPTION

`hcam` must specify a CCD camera that has been successfully opened by PVCAM.

This function uses the input string *`script`* to create a control program for *`hcam`*. The script controls all aspects of one or more exposures, including clearing the CCD, opening and closing the shutter, shifting the parallel register (or registers, in the case of frame transfer CCDs) and reading out the pixels. The script is also capable of coordinating with external triggers and triggering external hardware.

This function compiles the script into a form that is appropriate for control-ling *`hcam`*. During that compilation, it determines how many pixels will be collected. The total number of bytes is returned in *`stream_size`*; and the user must allocate a *`pixel_stream`* array of at least this size before running the script (see `pl_exp_start_script`).

During compilation, the scripts display functions are also examined. These functions specify how the resulting data is decomposed into individual images.

After this function has successfully returned, the caller is assured that the script is accepted and properly processed, the corresponding instructions have been transmitted to the camera (which is now ready to go), and the display instructions have been processed. Data collection may now start by calling `pl_exp_start_script` function.

RETURN VALUE

TRUE for success, FALSE for a failure. Failure sets pl_error_code.

SEE ALSO

`pl_exp_start_script(101), pl_exp_abort(0), pl_exp_check_status(0)`

NOTES

Most of the complexity of this function is contained in the scripting language. Read this manual for a more complete description of this function.

NAME

`pl_exp_start_script` – begin exposing, return immediately.

SYNOPSIS

```
boolean
  pl_exp_start_script(int16 hcam, void_ptr pixel_stream)
```

DESCRIPTION

`hcam` must specify a CCD camera that has been successfully opened by PVCAM.

This function is the companion function to `pl_exp_setup_script`. `pl_exp_setup_script` must be called first to define the exposure and program this information into the camera. After that, `pl_exp_start_script` may be called one or more times. Each time, it will trigger the script to begin executing from the start, and then immediately return. (Since a script may have been aborted in the middle, leaving the camera in an unknown state, well written scripts will usually force the camera to a known state at the start of the script.)

Exposure progress is monitored through `pl_exp_check_status`. The next script can be started as soon as the readout is finished or an abort is performed (`pl_exp_abort`).

The user must allocate *`pixel_stream`*, an appropriately sized memory buffer for data collection. The buffer must be at least *`stream_size`* bytes large, where *`stream_size`* is the value returned from `pl_exp_setup_script`. In addition, this memory needs to be page locked or similarly protected on virtual memory systems. These requirements are system specific.

There is a special case for users who want to use their own frame grabber (with an appropriately equipped camera). If a null pointer is passed in for *`pixel_stream`*, `pl_exp_start_script` will assume that the user is routing the data to a frame grabber or other device under user control. Under those conditions, `pl_exp_start_script` will initiate the exposure, but will not attempt to handle incoming data.

RETURN VALUE

TRUE for success, FALSE for a failure. Failure sets pl_error_code.

SEE ALSO

`pl_exp_setup_script(101), pl_exp_abort(0), pl_exp_check_status(0)`

NOTES

This is programmed as macro call `pl_exp_start_seq`. This script function exists to provide a complete and coherent conceptual model for the sequence style exposure functions. From the API level, it should always appear that an exposure style is self-contained and complete, and that each exposure style is completely independent from every other exposure style. Underneath the API, though, code will be reused as needed.

NAME

`pl_exp_uninit_script` – unitialize the scripting library.

SYNOPSIS

```
boolean
   pl_exp_uninit_script( void )
```

DESCRIPTION

This function undoes the preparations done by `pl_exp_init_script`. After executing this function, scripting functions may no longer be called.

RETURN VALUE

TRUE for success, FALSE for a failure. Failure sets pl_error_code.

SEE ALSO

`pl_pvcam_init(2), pl_pvcam_uninit(2), pl_exp_init_script(101)`

NOTES

This function must be explicitly called before calling `pl_pvcam_uninit`.

You must also uninitialize the Class 3 library (`pl_exp_uninit_seq`) whenever you call `pl_exp_uninit_script`. Order is not important. You may call either function first, but both of them need to be performed before calling `pl_pvcam_uninit`.

Decoding

Decoding (defined as decoding the pixel stream to display one or more images) must be simple enough to easily use, yet, powerful enough to display most experiments. To satisfy both needs, the scripts can contain any number of pixel display statements. Each of these statements must specify a rectangular array of pixels to be decoded and displayed. In addition, these instructions can be contained inside of loops, making them much more powerful and flexible.

Decoding the ICL

To decode the ICL, the scripting functions process, whitespace and the following commands:

```
script_begin();
loop_begin(loop_count);
pixel_display(x,y);
loop_end() ;
script_end(contin_clear);
```

Particular attention is paid to processing display commands inside of loops and loops inside of loops (loop nesting).

Once the script has been processed (through the call to `pl_exp_setup_script`), all display instructions have been examined and recorded. Application programmers can obtain the total number to display from the `pl_exp_setup_script` command. More detailed information is available from the `pl_exp_display_script` command.

Image Display

The image display method depends on the application. You should perform the extraction and display tasks using the same method you are currently using to process the pixel stream and display images with conventional PVCAM. In other words, if you already have a working application, you need to apply the same techniques used in that application to the image display. (See `pl_exp_display_script` for more information.)

Although the decoding and display must logically match the experimental setup, the `pixel_display` calls do not have to be in a one-to-one correspondence to the `pixel_readout` calls. The example scripts (see the *Example Scripts* section) illustrate situations where the readout and display are handled in very different ways inside the same script.

While the individual calls don't need to match, the total number of pixels collected must match the total number of pixels being displayed. The script processor (`pl_exp_setup_script`) checks for this matching and generates an error if there is any disagreement.

For more information concerning image display, please consult the `pl_exp_setup_script` and `pl_exp_display_script` functions.

This page intentionally left blank.

Chapter 3. Advanced CCD Theory

Introduction

The charge-coupled device (CCD) is the imager of choice for use in quantitative image acquisition systems. This chapter familiarizes you with CCDs and the terminology used in describing them. It also discusses CCD performance characteristics. If these are already familiar topics, you may still find this chapter useful as a refresher or a reference.

Theory of Operation

Most CCD imagers are made from silicon, because of its properties when exposed to electromagnetic radiation in the visible spectrum.

In crystalline silicon, each atom is covalently bonded to its neighbors. Incident photons that penetrate the lattice can break these bonds and generate electron-hole pairs. Silicon becomes transparent at 1000–1100 nm and is opaque to light at wavelengths shorter than 400 nm. The photon-produced electronic charge is proportional to the incident light between these wavelength limits.

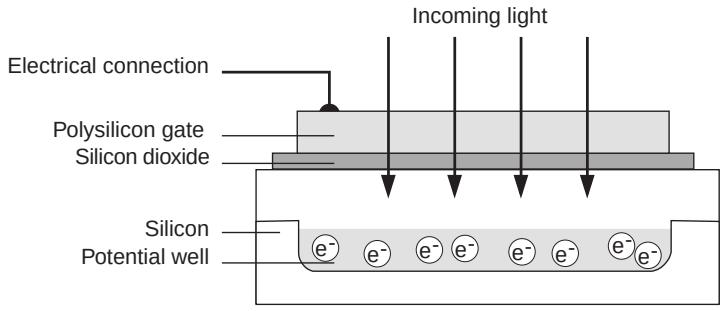
However, charge can be created by other energy sources. High-energy particles, x-rays, and cosmic rays can break many thousands of bonds. Excessive exposure can damage the crystal lattice.

Bonds can also be broken by thermal agitation. The rate of electron-hole pair generation due to thermal energy is dependent on temperature and can be reduced arbitrarily through cooling. The unwanted charge produced by thermal agitation is called dark current, because it is produced in the absence of light.

Potential Wells

To measure the electronic charge produced by incident photons, there must be a means of collecting the charge. The figure *A Potential Well* illustrates the concept.

A Potential Well



A thin layer of silicon dioxide is grown on a section of silicon and a conductive gate structure is applied over the oxide. Applying a positive electrical potential to the gate creates a depletion region where the free electrons generated by incoming photons can be stored. A potential well collects electronic charge from any source until the well is filled. Practical potential well capacities range up to a million electrons. Depletion depths range from a few micrometers up to tens of micrometers in specially prepared silicon.

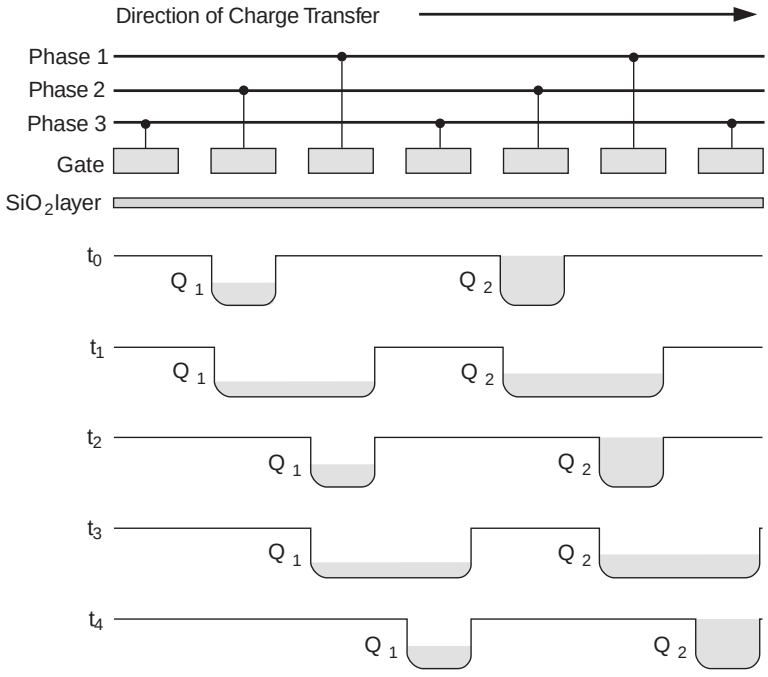
Electrons freed by thermal agitation and by high-energy particles are indistinguishable from those generated by photon interaction. These dark electrons can have an adverse effect on the detection limits for photon-induced charge.

Charge Transfer

The charge collected in a potential well must be brought to an output amplifier. When a series of oxide and conductive gate structures is fabricated with multiple phases, potential wells can be propagated through a silicon sheet.

This charge transfer concept is essential to understanding charge-coupled devices. When an appropriate sequence of potentials is applied to the gates, the potential wells are propagated in the direction shown in the figure *Charge Transfer*. Any charge that has been collected is carried along in the wells, and the charge packets in each potential well remain separated. Charge packets can be transferred thousands of times without significant loss of charge.

Charge Transfer



The illustration is simplified to emphasize the concept. To ensure effective charge transfer, charge propagation actually occurs in a channel buried just below the surface, where there is no interference from interface states. The gates actually overlap, to create the drift field required for efficient charge transfer.

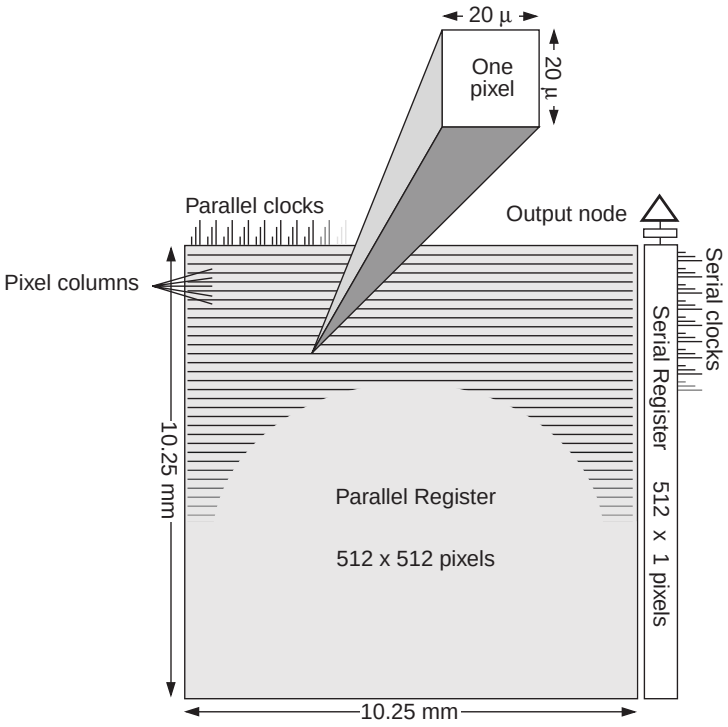
Classical CCD Implementations

The one-dimensional charge transfer concept can be extended to two dimensions. Buried channels arranged in parallel establish columns; charge cannot migrate between columns. Along each column, charge is contained in individual potential wells by a multiphase gate structure, as discussed for the one-dimensional case. The result is a two-dimensional array of independent potential wells.

A Typical CCD Imager shows a stylized CCD imager. The large square array is the parallel register. In this example, the parallel register contains 262,144 independent potential wells, called picture elements or pixels. Each pixel is

20 µm square and contains a single charge packet.

A Typical CCD Imager



The serial register is along one side of the parallel register, perpendicular to the pixel columns. The serial register is itself a one-dimensional CCD. It may be masked so that incident photons cannot create charge.

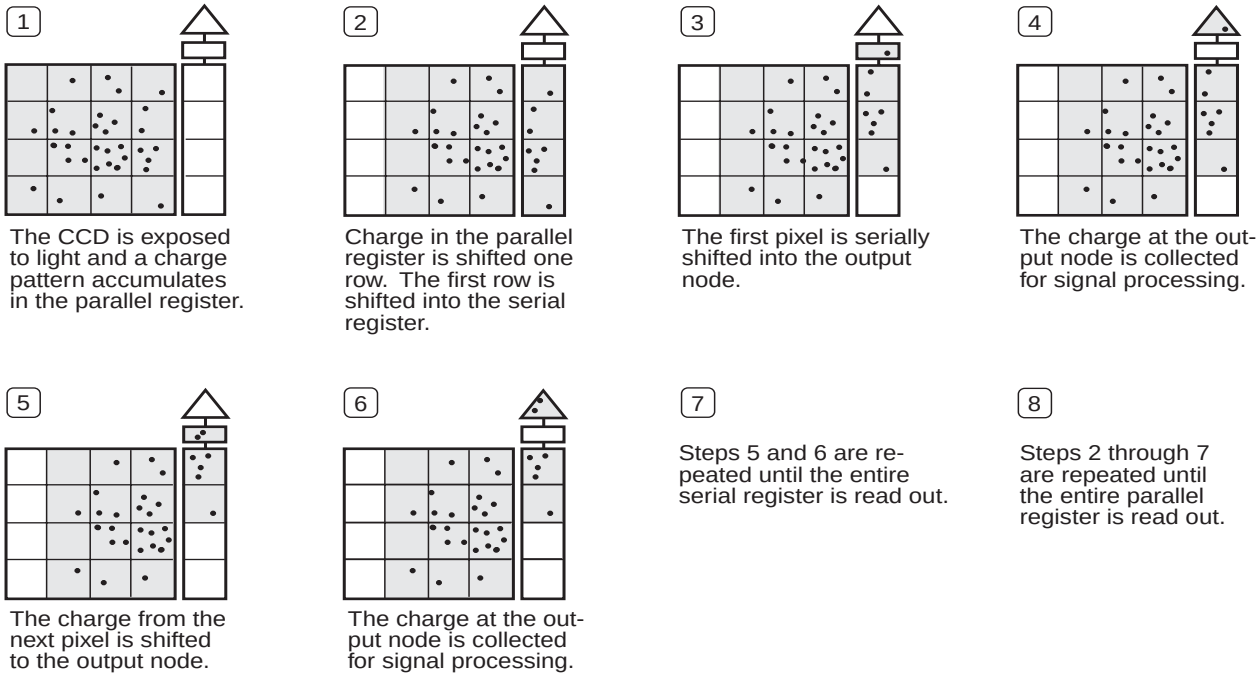
Many CCD formats and aspect ratios are available; the choice of CCD is dictated by the application.

When a CCD imager is exposed to light, charge accumulates in the potential wells of the parallel register. Charge can accumulate over an extended period of time. The total amount of charge is proportional to the product of the light intensity and the exposure time. (A charge accumulation is often called an integration.) The complete charge pattern corresponds to the focused image.

CCD Readout

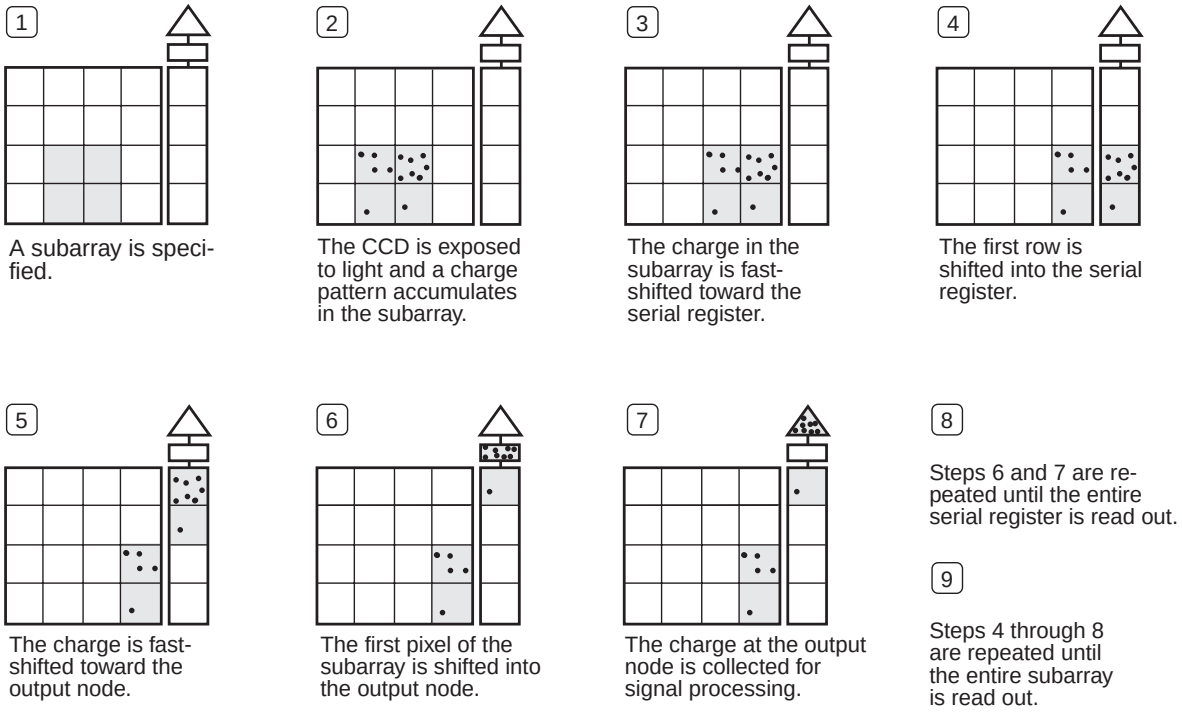
The standard CCD readout sequence is shown in *CCD Readout*. After an integration, a programmed sequence of changing gate potentials causes all charge packets stored in the parallel register to be shifted one pixel towards the serial register; the first charge packet in each column is shifted into the serial register. Once in the serial register, charge packets are individually shifted toward the output amplifier. The output amplifier produces a signal proportional to the charge in each packet. After the serial register is emptied of charge, a second row of charge packets is shifted in from the parallel register. The process continues until all charge has been shifted out of the parallel register.

CCD Readout



All CCD imagers depend on the efficient transport of charge from the photosites to the output amplifier. Because the charge from wells located far from the output amplifier must undergo many hundreds of transfers, the charge transfer efficiency (CTE) is important. A scientific-grade CCD imager exhibits a CTE of 0.99999, where 1.0 is perfect. CTE is of special concern at low charge levels where a small loss of charge can cause significant degradation of the image.

Subarray Readout



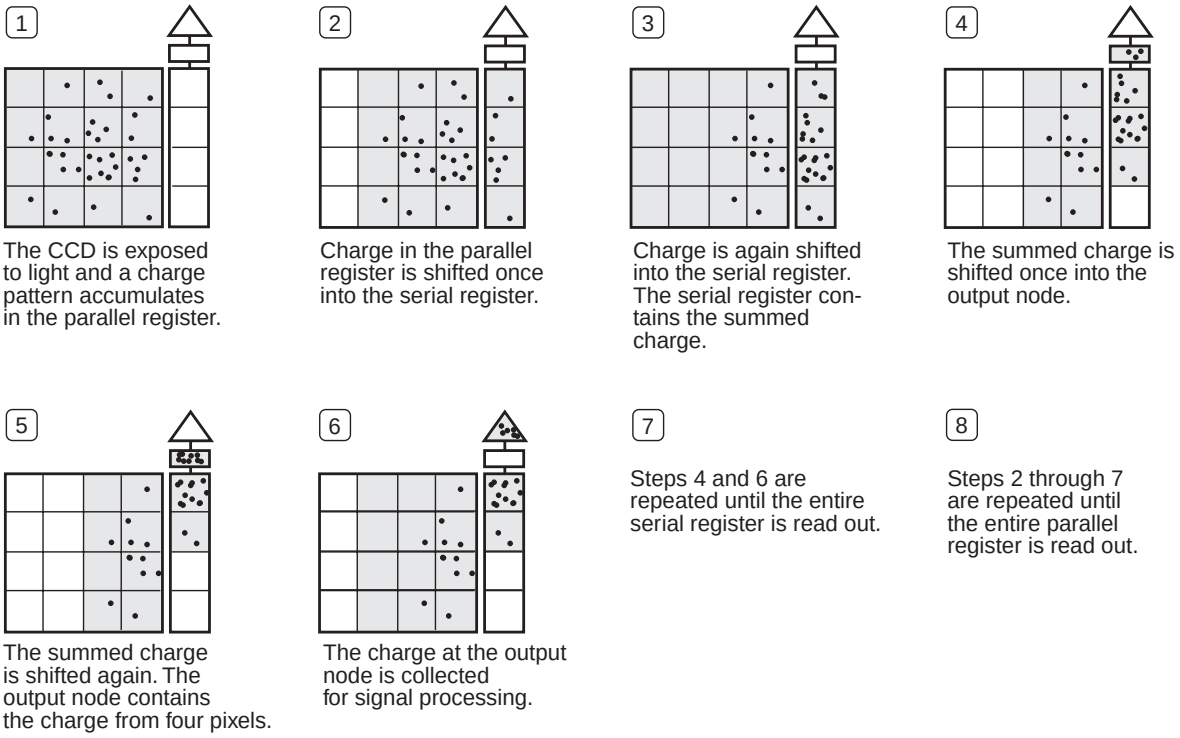
Subarrays It is not necessary to read out all of the pixels on the parallel register. One or more subarrays on the parallel register may be specified and read out, with a resultant reduction in pixel count and increase in readout speed. *Subarray Readout* illustrates the subarray readout concept.

Binning Binning is a technique of combining charge from adjacent pixels during the readout process. Binning improves the signal-to-noise ratio and extends the dynamic range of the CCD imager, but at the expense of spatial resolution. Binning is thus most useful in applications where resolution in one or both axes is not of primary concern. Because binning reduces the number of pixels to be processed and digitized, readout speed is also increased.

As shown in *Binned Readout*, on page 34, charge is collected in the normal fashion, but the readout is programmed differently. With parallel binning, when charge is shifted from the parallel register into the serial register, charge is accumulated from two or more rows before serial shifting begins. With serial binning, two or more charge packets are similarly accumulated in the output node before the charge is digitized and read out.

Binning is specified by a binning factor: the number of pixels to be combined on the CCD. The example shows 2 x 2 binning. Any combination of parallel and serial binning factors is possible. The default values of 1 provide no binning.

Binned Readout



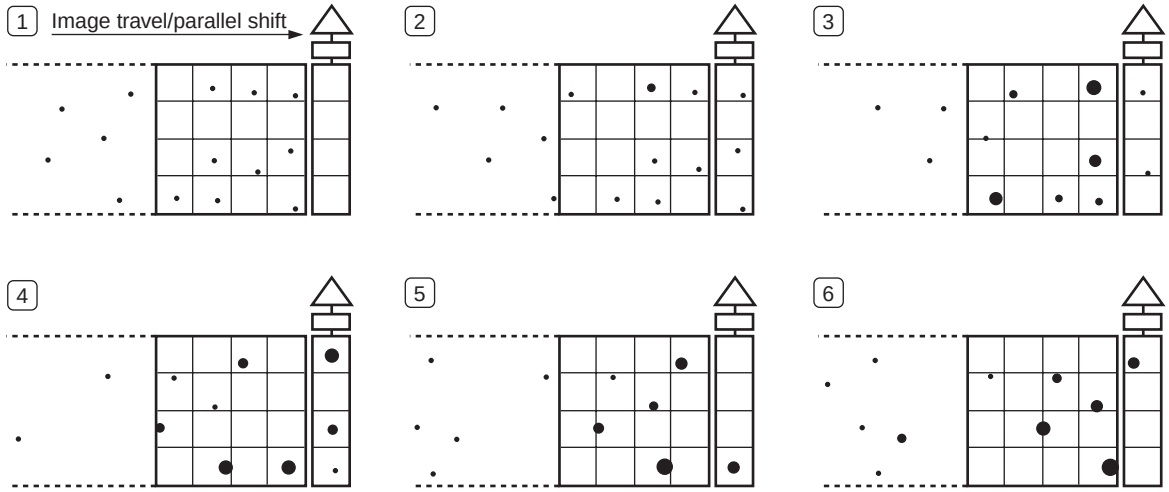
Time Delay Integration

Time delay integration (TDI) is an integration and readout mode that allows the acquisition of long swaths of a moving image.

The figure *Time Delay Integration* on page 35 illustrates the TDI principle. A moving image is focused on an unshuttered CCD imager. The parallel register is clocked in step with image motion, so that charge packets always correspond to the same image region as they move across the parallel register. Charge accumulates and signal strength increases as the pixels approach the serial register. When pixels reach the serial register, they are transferred out, digitized, and stored in the normal fashion. The exposure time for each pixel is exactly the length of a full parallel shift sequence, which is determined by the velocity of the scene.

Compared to a simple exposure, TDI increases sensitivity in proportion to the number of rows in the parallel register. Although the TDI technique was first used by astronomers to take images of passing star fields, it has also been used to acquire swath images from airplanes and satellites, and is very effective for inspecting articles on a moving conveyor belt. TDI is normally used with scenes moving at a constant velocity, but other variations are possible.

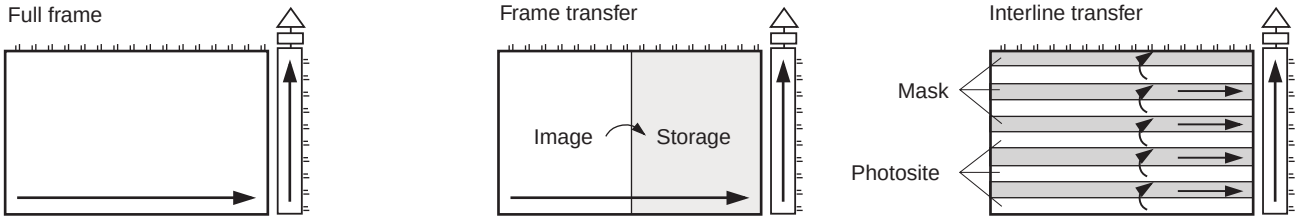
Time Delay Integration



CCD Architectures

Three types of CCD imagers are used for quantitative electronic imaging. CCD Architectures illustrates basic CCD structures that are in current use.

CCD Architectures



Full Frame

The standard full-frame CCD imager has the simplest architecture. The parallel register is used for photon exposure, charge integration, and charge transport. A shutter is used to control the exposure and to block light during readout, preventing charge smearing.

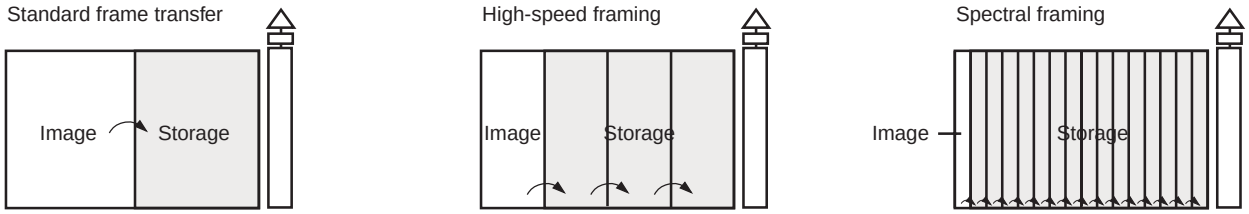
Frame Transfer

The frame-transfer CCD imager has a parallel register divided into two distinct areas. The area at the far end of the parallel register is the image array, where images are focused and integrated. The other area, the storage array, is identical in size and is covered with an opaque mask to provide temporary storage for collected charge.

After the image array is exposed to light, the entire electronic image is rapidly shifted to the storage array. While the masked storage array is read, the image array may integrate charge for the next image. A frame-transfer CCD imager can operate continuously without a shutter at a high rate.

There are some special adaptations of frame-transfer devices, shown in *Frame Transfer Variations*.

Frame Transfer Variations



High-Speed Framing

The frame transfer concept can be extended to multiple frames by masking most of the parallel register and using only a small region as the image array. A scene is focused on the image array and a high-speed shutter or strobe light is used to time the exposure. After each exposure, charge from the image array is quickly shifted under the mask and a new image can be acquired. Once the parallel register is filled with images, it is read out.

Because fewer rows are clocked to shift the image array into storage, this mode works much faster than standard frame transfer.

The high-speed framing rate need not be constant, but may be varied in response to the needs of the observation being made. High-speed spectra of a decaying phosphor, for example, may be obtained with decreasing time resolution in response to the exponential decay time of the emitted light.

Spectral Framing

In a mode particularly suited for spectroscopy, the CCD is masked so that only a single row of the parallel register is exposed. In this mode, one-dimensional line images can be acquired at a very high speed until the parallel register is filled up.

Spectral-framing CCDs are used in time-resolved spectroscopy. An observation could consist of hundreds of individual spectra, distributed over time. The CCD clocks are controlled by computer, so non-linear time bases are possible.

Interline Transfer

The interline-transfer CCD has a parallel register that has been subdivided so that the masked storage area fits between columns of exposed pixels. The electronic image accumulates in the exposed area of the parallel register, just as it does in the frame-transfer CCD. At readout, the entire image is shifted under the interline mask. The masked pixels are read out in a fashion similar to the full-frame CCD.

To enhance the interline transfer CCD's sensitivity, micro lenses are applied directly to the CCD's surface. The micro lenses cover the entire pixel and part of the mask on either side of the pixel. In this way, the micro lenses help to focus light to the CCD's parallel register by funneling photons to active pixels rather than allowing them to bounce off of the masks.

CCD Camera Implementations

To take advantage of the high performance a CCD has to offer, special slow-scan or still-imaging cameras have been designed to operate at a significantly lower speed than conventional video cameras. These cameras bring together several diverse technologies: high-performance signal processors, solid-state coolers, precision digitizers, and high-speed digital controllers. The benefits of slow-scan readout are ultra-low noise, maximum CCD performance, and photometric precision in the image data.

A precision analog processing circuit and analog-to-digital converter are employed to amplify and digitize the CCD output signal. CCD readout may take from one tenth of a second up to several seconds, as each pixel is digitized with up to 16-bit precision. The digitized electronic image read out from the CCD can be stored in a computer's memory. Slow-scan camera systems produce large quantities of data. A 2000 x 2000 pixel CCD with a dynamic range of 20,000:1 requires eight megabytes of storage for each image.

Cooling the CCD reduces dark current to negligible levels, allowing exposure times of up to hours in duration. To achieve the highest possible sensitivity, astronomers cool the CCD with liquid nitrogen, eliminating the dark current produced by thermal generation at warmer temperatures. High energy physicists, on the other hand, use CCDs in ultra-high-speed cameras to observe transient phenomena where dark current is not relevant.

A conventional shutter can be used to acquire exposures as brief as a few milliseconds or as long as an hour; a microchannel plate image intensifier can be used to gate exposures of a few nanoseconds.

Resolution

The resolution of a CCD camera is determined by the geometry of the specific CCD in use. The CCD pixels set the limit of resolution. In scientific-grade CCDs, a pixel varies in size from a few micrometers up to 48 µm; the total imaging area is 1 to 24 cm^2. In order to avoid aliasing, moiré, or beat frequencies, the magnification must be chosen so that at least two CCD pixels cover a desired resolution element in the image plane.

There is no dead space between pixels. Charge generated by photons striking the CCD between pixels diffuses to the nearest potential well. This is referred to as "fill factor."

Sensitivity

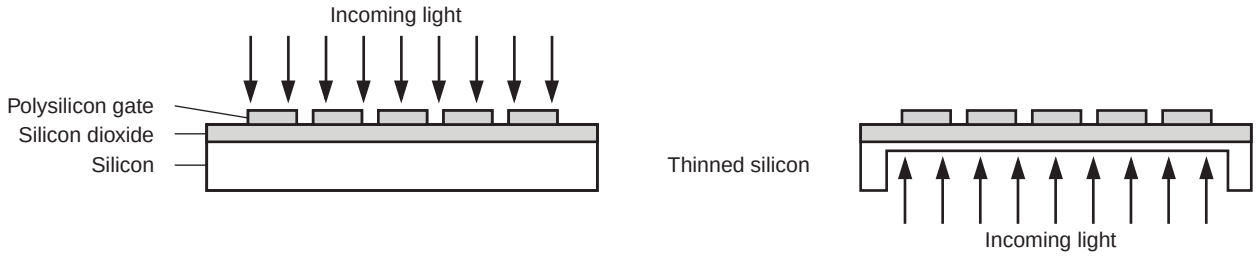
The sensitivity of a CCD imager to light is determined by system noise level (discussed later in this chapter) and quantum efficiency.

Quantum efficiency measures the sensor's efficiency in generating electronic charge from incident photons. Electron-hole pairs are produced by photons in the region from 400 to 1100 nm. Within the visible spectrum, the photon to electron conversion factor is less than unity and it varies as a function of wavelength. At a given wavelength, the creation of charge from incident light is intrinsically linear.

Spectral Response Light normally enters the CCD through the gates of the parallel register. These gates are made of very thin polysilicon that is reasonably transparent at long wavelengths but becomes opaque at wavelengths shorter than 400 nm. Thus, at short wavelengths the gate structure attenuates incoming light.

Thinning With acid etching techniques, CCDs can be uniformly thinned to approximately 10 µm, and an image can be focused on the backside of the parallel register where there is no gate structure. These thinned, or backside-illuminated, CCDs exhibit high sensitivity to photons from the soft x-ray to the near-infrared regions of the spectrum. *Thick and Thinned CCDs* illustrates the structures of both kinds of device.

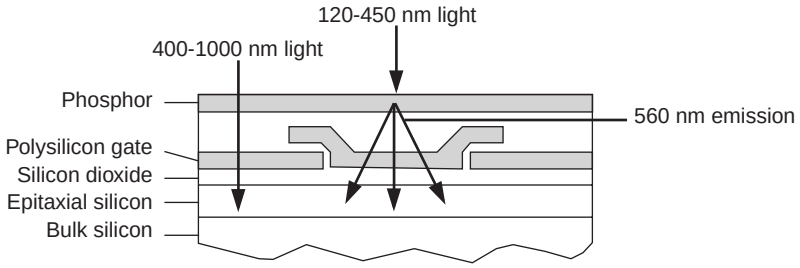
Thick and Thinned CCDs



Down Converters CCD spectral response can also be extended with the use of a light-emitting phosphor called a down converter. A down converter absorbs light in the ultraviolet range of the spectrum and re-emits it in the visible range.

A CCD with a Down Converter demonstrates the use of the phosphor down converter Metachrome® II in conjunction with a front-illuminated CCD.

A CCD with a Down Converter



The phosphor absorbs photons with short wavelengths and emits photons with a 560 nm wavelength. These photons pass through the polysilicon gates into the photon-sensitive region of the CCD. The down converter is transparent between 400 and 1000 nm, so CCD performance is not affected at these longer wavelengths.

Down converters exhibit very high quantum efficiency and, when properly processed and applied to CCDs, can produce a significant improvement in UV sensitivity.

Fiberoptics

In most applications, the CCD is illuminated by light from an imaging source such as a telescope, spectrograph, or microscope. There is a class of applications, however, in which coherent fiberoptic bundles can be used in place of imaging optics.

A fiberoptic bundle comprises millions of glass fiber strands, each of which acts as an individual light pipe. Each strand has a glass cladding that isolates it from neighboring strands to prevent crosstalk. The optical fibers are carefully stacked and fused together under heat and pressure to form a solid, rigid coherent optical transmission element. The fiber boules that are produced in this process may be reheated and drawn into fiber tapers to meet the needs of specific applications. Fiberoptic bundles may be twisted 180° to form image inverters.

A fiberoptic bundle may be bonded directly to a CCD, effectively translating the image plane from the surface of the CCD out to the surface of the fiberoptic bundle. The interface between the CCD and the fiberoptic must be made with great care to prevent damage to the delicate CCD surface.

Fiberoptic–coupled CCDs are often used in conjunction with other optical devices, such as image intensifiers and streak tubes, equipped with matching fiberoptic outputs. The high light collection efficiency provided by this approach warrants the additional complexity in fields such as radiography and electron microscopy, where fiberoptic–coupled CCDs are used to directly image phosphor screens.

Sources of Noise

All electronic circuitry generates undesirable noise. The effect of this noise on performance is described by the signal-to-noise ratio (SNR). Photon noise, preamplifier noise, and dark current noise are the three primary sources of noise in a CCD camera.

Photon Noise

Photon noise, also known as photonic or photon shot noise, is a fundamental property of the quantum nature of light. The total number of photons emitted by a steady source over any time interval varies according to a Poisson distribution. The charge collected by a CCD exhibits the same Poisson distribution, so that the noise is equal to the square root of the signal. Photon noise is unavoidable and is always present in imaging systems; it is simply the uncertainty in the data.

Preamplifier Noise

Preamplifier noise, also called read noise, is generated by the on-chip output amplifier. This noise can be reduced to a few electrons with careful choice of operating conditions.

Dark Current Noise

Dark current, or thermally generated charge, can be measured and subtracted from data, but its noise component cannot be isolated. Dark current noise is a particular concern in low-light applications.

To reduce dark current, CCDs can be chilled to approximately –50°C with thermoelectric cooling. For extremely long exposures, liquid nitrogen is used to cool the CCD to as low as –120°C. At lower temperatures, CCD performance may be degraded due to poor charge transfer efficiency.

MPP Operation

Multi-pinned-phase (MPP) or inverted operation reduces the rate of dark current generation by a factor of 20 or more and thus relaxes CCD cooling requirements to the level where a thermoelectric cooler is sufficient for most applications.

Most of the dark current in a CCD is generated by interface states at the silicon–silicon dioxide interface just below the parallel gate structure. In MPP mode, this dark current component is significantly reduced by biasing all of the parallel register gates into inversion. However, this causes the potential wells essential for operation to disappear, allowing charge to spread up and down columns. Efficient CCD action can be ensured by processing CCDs with a built-in potential step that restores the potential wells when the parallel gates are biased at the same voltage. Only CCDs thus processed can be operated in inverted mode.

Tradeoffs

In a given situation, the available light level determines the integration time required to arrive at an acceptable SNR. Acceptable SNRs vary with the application; the tradeoffs between light level and integration time must be considered for each circumstance.

When the light level is high enough that photon statistics are the dominant source of noise, preamplifier noise and dark current are not relevant. The CCD data are said to be photon noise limited. Under low-light conditions, where preamplifier noise exceeds photon noise, the CCD data are said to be preamplifier noise limited. When long integration times are used, it is important to ensure that the noise from dark current does not exceed preamplifier or photonic noise from the signal.

Two examples give some insight into the tradeoffs required to maximize the SNR:

- Solar astronomy is a typical high-light-level CCD application. For this application, it is important to detect small fluctuations in intensity over the area of the sun as a function of time. Because the light source is very bright, a slow-scan CCD camera always operates under photon noise limited conditions.
- Low-light-level conditions, such as those encountered in fluorescence microscopy, present a different problem. For this application, the photon flux is typically low and the excitation exposure must be kept short to avoid bleaching. CCD sensitivity and preamplifier noise are extremely important. If the CCD has a preamplifier noise of 10 electrons, the image data are preamplifier noise limited when the number of photoelectrons in a pixel is less than 100. For signals above 100 electrons, the data are photon noise limited.

Additional Reading

The following articles provide more information on CCDs and their applications.

Aikens, Richard S. *Charge-Coupled Devices for Quantitative Electronic Imaging*. Tucson, Arizona: Photometrics, Ltd., 1991.

Aikens, Richard S., Patrick M. Epperson, and M. Bonner Denton. "Techniques for Operating Charge Coupled Devices (CCD's) in Very High Speed Framing Mode." *Proceedings of the Society of Photo-Optical Instrumentation Engineers* 501, 1984.

Dereniak, E.L. and Crowe, D. *Optical Radiation Detectors*. New York: John Wiley and Sons, 1984.

Janesick, James, and Morley Blouke. "Sky on a Chip: The Fabulous CCD." *Sky and Telescope*, September, 1987.

Janesick et al (1989). "Charge-Coupled Device Pinning Technologies." *Proceedings of the Society of Photo-Optical Instrumentation Engineers* 1071-15, 1989.

Kristian, Jerome, and Morley Blouke. "Charge-Coupled Devices in Astronomy." *Scientific American*, October, 1982.

Sweedler, Jonathan V., Robert B. Bilhorn, Patrick M. Epperson, Gary R. Sims, and M. Bonner Denton. "High-Performance Charge Transfer Device Detectors." *Analytical Chemistry* 60: 282A, 1988.

This page intentionally left blank.

Index

W-Z

BENELUX
Roper Scientific, BV
Ir. D.S. Tuijnmanweg 10
4131 PN VIANEN, Netherlands
tel: 31.347.324989
fax: 31.347.324979
email: mailto@roperscientific.com

FRANCE
Roper Scientific, SARL
Z.I. Petite Montagne Sud
4, rue de l'Oisans - C.E. 1702
91017 Evry Cedex, France
tel: 33.160.86.03.65
fax: 33.160.86.07.09
email: princeton.instruments@wanadoo.fr

GERMANY
Roper Scientific, GmbH
Rosenheimer Landstr. 87
D-85521 Ottobrunn, Germany
tel: 49.89.660.779.3
fax: 49.89.660.779.50
email: mail@roperscientific.de

JAPAN
Nipon Roper, K.K.
D-10E 1-3 Nakase,
Mihama-ku, Chiba-shi
Japan 261-8501
tel: 81.43.274.8022
fax: 81.43.274.8023
email: sales@roper.co.jp

USA
Roper Scientific – Arizona
3440 East Britannia Drive
Tucson, Arizona 85706
tel: 520.889.9933
fax: 520.295.0299
email: cservice@roperscientific.com

USA
Roper Scientific – New Jersey
3660 Quakerbridge Road
Trenton, New Jersey 08619
tel: 609.587.9797
fax: 609.587.1970
email: techsupport@roperscientific.com

57-039-001 Rev C1